Departement Informatica en Informatiekunde, Faculteit Bètawetenschappen, UU.
In elektronische vorm beschikbaar gemaakt door de TBC van A–Eskwadraat.
Het college INFOIMP werd in 2007/2008 gegeven door Jeroen Fokker.

Imperatief Programmeren, derde deeltentamen (INFOIMP) 2 november 2007

Opgave 1 *(20 punten)*

Voor elk punt (x, y) van het platte vlak, waarbij x en y reëele getallen zijn, kan een bijbehorend getal worden bepaald: het zogeheten "mandelgetal". Om het mandelgetal te kunnen uitrekenen, bekijken we eerst de volgende functie, die punten (a, b) van het vlak transformeert naar andere punten:

$$f(a, b) = (a * a - b * b + x, 2 * a * b + y)$$

Let op: deze functie transformeert het punt (a, b), maar in de berekening speelt ook de waarde van x en y, dat is het punt waarvan we het mandelgetal willen bepalen, een rol.
Deze functie f nu, passen we toe op het punt $(a, b) = (0, 0)$. Op het punt dat daar uitkomt, passen we nog eens de functie f toe, enzovoorts enzovoorts. We stoppen pas met toepassen van f als het resultaatpunt een afstand van meer dan 4 tot de $(0, 0)$ heeft. Het mandelgetal is nu gelijk aan het aantal keren dat f is toegepast.
Voor sommige punten (x, y) is dat het meteen al zo, en is het mandelgetal dus gelijk aan 1. Voor andere punten duurt het langer: die hebben een groter mandelgetal. Er zijn ook punten waarbij je f kan blijven toepassen, zonder dat de afstand tot de oorsprong ooit meer dan 4 wordt. Voor di epunten stellen we het mandelgetal op 100.

a) Schrijf een methode `mandel` die het mandelgetal uitrekent van het punt waarvan de coördinaten als parameter worden meegegeven.

b) Schrijf het ontbrekende stuk van de methode `paint`, die de punten op het scherm zwart kleurt die een *even* mandelgetal hebben. De gedeclareerde schaal moet worden gebruikt zo dat het plaatje wordt getoond voor x en y tussen 0 en 4.

```
class Mandel extends Applet
{   double schaal = 0.01;
    public void paint(Graphics g)
    {   int x, y;
        for (x=0; x<400; x++)
        {
            .......... opgave b ..........
        }
    }
    .......... opgave a ..........
}
```

Opgave 2 *(40 punten)*

Let op: als er meerdere vragen gesteld worden: elke vraag apart beantwoorden. Houd het antwoord kort: een of twee regels per vraag kan al genoeg.

a) Geef met een paar regel code aan hoe je in de methode `paint` de te gebruiken lijndikte kunt instellen.

b)
- Wat is het voordeel van een `ArrayList` boven een gewone array?
- Net als andree implementaties van `Collection` kun je zo'n `ArrayList` *generic* maken. Wat is daarvan het voordeel boven een niet-generic exemplaar?

c) Als je de Swing-toolkit gebruikt, kun je een menu op elke gewenste plaats in de gebruikersinterface neerzetten. Bij de AWT-toolkit kan dit niet.

- Beschijf aan de hand van de klasse-hiërarchie waarom dit zo is.
- Waarom was het voor de auteurs van AWT niet mogelijk om deze vrijheid te bieden, en voor de dauteurs van Swing wel?

d) Wat is het verschil tussen het type `int` en het type `Integer`?

In welke situatie is het nodig om `Integer` te gebruiken in plaats van `int`?

e) Sommige klassen in de standaard-packages zijn `abstract` gemaakt, bijvoorbeeld klasse `Component` in de java.awt, en klasse `Reader` in java.io.

- Wat is de reden dat dit soort klassen abstract? (Beschrijf dit in het algemeen, het is niet nodig om specifiek op de genoemde voorbeelden in te gaan.)
- Sommige klassen die een interface implementeren zijn ook abstract, bijvoorbeeld de klasse `AbstractAction`. Wat is in dit soort gevallen vaak de reden dat zo'n klasse abstract moet zijn?

Opgave 3

(40 punten)

Het is in deze opgave toegestaan, maar niet per se noodzakelijk, om naast de gevraagde methodes ook nog extra hulpmethodes te schrijven.

Het programma in deze opgave leest een bestand in waarin d eligging en grootte van een aantal cirkels wordt vastgelegd. He tprogramma toont deze cirkels op het scherm. onderaan het scherm is een schuifbalk aanwezig.

De gebruik kan in het scherm met de cirkels klikken. Als hij/zij daarbij een bestaande cirkel raakt, verandert de diameter van die cirkel: die wordt zo groot als de schuifbalk op dat moment aangeeft. Als de gebruik buiten de bestaande cikels klikt, ontstaat gecentreerd op die plek een nieuwe cirkel met een diamter zoals ingesteld met de schuifbalk. Er is geen grens op het aantal cirkels.

In de bijlage is een deel van het programma gegeven. Het bestaat uit vier klassen: `Test`, `Kaart`, `Cirkel` en `Punt`. Een aantal onderdelen moet nog worden ingevuld.

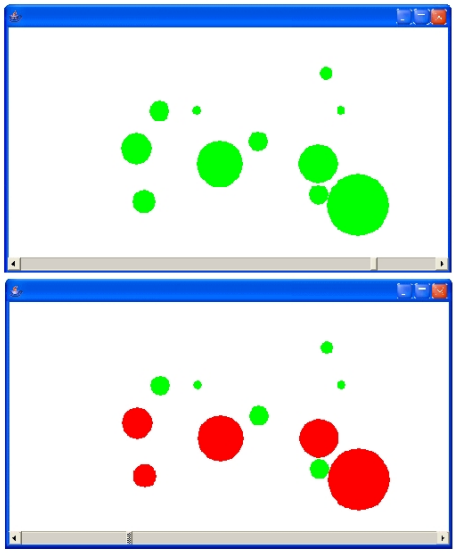
Let op dat sommige variabelen `private` zijn gedeclareerd: die mogen van buiten de klasse niet worden gebruikt.

a) Schrijf de methode `raak` in de klasse `Cirkel`, die oplevert of de paramters een punt binnen de cirkel aanduiden.

b) Schrijf de declaraties van de variabelen die nodig zijn bovenin de klasse `Kaart` en de constructormethode van deze klasse.

c) Schrijf het ontbrekende deel van de methode `lees` van de klasse `Kaart`, die de cirkels inleest vanuit een bestand waarvan de naam gegeven is. De regels van dat bestand bevatten steeds 3 getallen (de x- en y-coördinaat van het midden en de diameter), dus regels zoals

```
250 110 20
280 180 50
```



d) Geef de invulling van methode `welke` en schrijf de methode die verder nodig is om het beschreven muisklik-gedrag te krijgen.

e) Schrijf de methode `paint` die alle cirkels in beel brengt. D ekleur van de cirkel moet zijn zoals de gegeven methode `kleurVan` dat voorschrijft.

f) We gaan het programma nu aanpassen. In de klasse `Test` wordt het type `Kaart` in de declaratie en initialisatie van de variabele k vervangen door het type `Kaart2`. Schrijf de klasse `Kaart2` zodat het programma zich als volgt gaat gedraagen: alle cirkels die groter of gelijk zijn aan de waarde die met de schuifbalk is ingesteld, moeten in rood worden getekend.

In de klasse `Kaart2` moet zo veel mogelijk van het werk dat al in de klasse `Kaart` is gedaan worden hergebruikt; het mag dus niet opnieuw worden opgeschreven.

Bijlage bij opgave 3*

```
 class Test extends Frame
        implements WindowListener, AdjustmentListener
{
    private Kaart k;
    private Scrollbar s;

    public Test()
    {   this.setSize(700,500);
        k = new Kaart();
        k.setParent(this);
        k.addMouseListener(k);
        s = new Scrollbar(Scrollbar.HORIZONTAL
                          , 20, 1, 1, 100);
        s.addAdjustmentListener(this);

        this.add(k, BorderLayout.CENTER);
        this.add(s, BorderLayout.SOUTH);
        this.addWindowListener(this);
        k.lees("cirkels.txt");
    }

    public int getWaarde()
    {   return s.getValue();
    }

    public bvoid adjustmentValueChanged(AdjustmentEvent e)
    {   k.repaint();
    }

    public static void main(String [] args)
    {   new Test().setVisible(true);
    }

    public void WindowClosing(WindowEvent e)
    {   System.exit(0);
    }

    public void windowOpened(WindowEvent e) {}
    public void windowClosed(WindowEvent e) {}
    public void windowActivated(WindowEvent e) {}
    public void windowDeactivated(WindowEvent e) {}
```

```
    public void windowIconified(WindowEvent e) {}
    public void windowDeiconified(WindowEvent e) {}
}

class Punt
{
    public int x, y;

    Punt(int x0, int y0)
    {   x = x0;
        y = y0;
    }
}

class Cirkel
{
    private int diam;
    private Punt loc;
    public Cirkel(Punt p, int d )
    {   loc = p;
        diam = d;
    }

    void setDiam(int d)
    {   diam = d;
    }

    int getDiam()
    {   return diam;
    }

    public void teken(Graphics g, Color c)
    {   g.setColor(c);
        g.fillOval(loc.x-diam/2, loc.y-diam/2, diam, diam );
    }

    static int kwadraat(int a)
    {   return a*a;
    }

    public boolean raak(int x, int y)
    {
        .......... opgave a ..........
    }
}

class Kaart extends Canvas implements MouseListener
{
    .......... opgave b ..........

    public void setParent(Test p)
    {   parent = p;
    }

    public void lees(String naam)
    {   try
        {
```

```
            .......... opgave c ..........
        }
        catch(Exception e)
        {   System.out.println(""+e);
        }
    }

    public Cirkel welke(int x, int y)
    {
        .......... opgave d ..........
    }

    public Color kleurVan(Cirkel c)
    {   Color k;
        .......... werking niet precies bekend ..........
        return k;
    }

    public void paint(Graphics g)
    {
        .......... opgave e ..........
    }

    public void mouseReleased(MouseEvent e) {}
    public void mouseClicked(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
}
```